TODO: Complete this page (TBS)

Overview

Resources allow users represent and reason about a wide range of resource types, including reusables(eg. a machine) and reservoirs(eg. a battery). Individual tokens (activities) can produce or consume resources (or both). Resources can have upper and lower limits, as well as bounds on instantaneous production/consumption and cumulative production/consumption.

Reasoning about resources involves two pieces:

1. **Profile:** The Profile computes the changes to the resource over time. Each production or consumption of the resource (called a transaction) affects the profile. Profile information includes the maximum and minimum possible values for the resource, and information about max/min production and consumption (both instantaneous and cumulative).
2. **Flaw and Violation Detectors (FVDetectors):** An FVDetector decides how to report flaws and violations given the resource profiles. For example, if one of the profile levels dips below the resources lower limit, is this a flaw, a violation, or neither?

There are a variety of Profile and FVDetector implementations available, and it is straightforward to implement your own.

(we discuss extending)

1. In NDDL, extend existing resource classes to get desired behavior.
2. In NDDL, specify what type of profile is used to represent maximum/minimum values over time.
3. In NDDL, specify what type of detector is used to report flaws and violation.
4. In configuration file (TODO: link) specify how

For example, here is a user-defined resource that extends unary resource, uses X and Y:

The following piece of the configuration file specifies how the built-in solver should handle the flaws reported by the Y:

Important Notes

- To have resource flaws and violations reported, you must XYZ?
-

Options

There are ?? possible profiles that can be used:

Combinations to Use and Avoid

Implementation Matrices

There are many possible pieces of data that can be computed by profiles and monitored by flaw/violation detectors. Here we show which ones are computed and monitored by the various profiles and detectors:

	TimetableProfile	GroundedProfile	FlowProfile	IncrementalFlowProfile
LowerLevelMin	Y	Y		
LowerLevelMax	Y	*(1)		
UpperLevelMin	Y	*(1)		
UpperLevelMax	Y	Y		
InstConsumptionMin				
InstConsumptionMax				
InstProductionMin				
InstProductionMax				
CumConsumptionMin				
CumConsumptionMax				
CumProductionMin				
CumProductionMax				

Possible New Features

Eventually, we hope to incorporate the following improvements (and bug fixes) into a future version of the Resources module:

- Non-constant upper/lower limits. For example, consider a pool of available cars that might get smaller (cars break) or larger (new cars bought) over time. The only way to represent this currently is with 'dummy' production/consumption events.
- Preferred value version of grounded profiles, so a preferred value (instead of the earliest value) could be used for grounding.
- A state resource, both for unary states (eg: on/off) and multi-state (eg: red/yellow/green). If you need a state resource immediately, ask about the hack that the DynamicEUROPA team uses.
- The GroundedProfile? does not treat instantaneous/cumulative production/consumption as 'grounded' but should.
- Re-architect flaw/violation detection so a user can pick and choose. For example, the closed-world assumption might be desired for violations, but not for flaws.
- The OpenWorldFVDetector treats flaws in a way that is not really related to the 'open-world' concept (it doesn't report flaws due to quantity flexibility). This behavior should be separated out; not necessary as part of open-world approach, and available in closed-world approach.

If you have a need for one of these listed features, please contact the EUROPA development team and we will attempt to fast-track support for that features.c